6 APPENDIX

6.1 ALGORITHMS

Algorithm 1: The **FLDmamba** Algorithm

Input: $\mathbf{X}$: (B, L, V);
Output: $\widehat{\mathbf{Y}}$: (B, L, V);

$U \leftarrow$ **FMamba**($\mathbf{X}$); // Step into FMamba algorithm 2
$U' \leftarrow$ **Mamba**($\mathbf{X}$); // Step into the Mamba algorithm 3
$U'' \leftarrow U' + U$;
$y' \leftarrow$ **FMamba**(U''); // Step into FMamba algorithm 2;
$y'' \leftarrow$ **Mamba**(U''); // Step into Mamba algorithm 3;
$Y \leftarrow \text{FFT}(y' + y'')$;
$Y \leftarrow \text{Linear}(Y)$;
$\widehat{\mathbf{Y}} \leftarrow \text{ILT}(Y)$; // Inverse Laplace Transform module
return $\widehat{\mathbf{Y}}$;

Algorithm 2: The **FMamba** Algorithm

Input: $\mathbf{X}$: (B, L, V);
Output: U:(B, L, V);

$\mathbf{X}' \leftarrow \text{RBF}(\mathbf{X})$;
for $p = 1, 2, ..., FMamba\ layers$ **do**
  $\mathbf{A}$: (V, N) $\leftarrow$ Parameter
  $\mathbf{B}$: (V, L, N) $\leftarrow s_B(\mathbf{X}')$
  $\mathbf{C}$: (B, L, N) $\leftarrow s_C(\mathbf{X}')$
  Δ: (B, L, N) $\leftarrow \tau_\Delta$(Parameter + $s_\Delta(\mathbf{X}')$)
  $\Delta' = \text{FFT}(\Delta)$
  $\Delta_F = \text{IFFT}(\Delta')$
  $\bar{\mathbf{A}}_F, \bar{\mathbf{B}}_F$: (B, L, V, N) $\leftarrow discretize(\Delta_F, \mathbf{A}, \mathbf{B})$
  $U^{(1)} \leftarrow \text{SSM}\ (\bar{\mathbf{A}}_F, \bar{\mathbf{B}}_F, \mathbf{C})(\mathbf{X}')$
  $U^{(2)} \leftarrow U^{(1)} \otimes \text{SiLU}(Linear(\mathbf{X}'))$
  $U \leftarrow Linear(U^{(2)})$
end
return U;

Algorithm 3: The **Mamba** Algorithm

Input: $\mathbf{X}$: (B, L, V);
Output: U':(B, L, V);

$\mathbf{X}' \leftarrow \text{RBF}(\mathbf{X})$;
for $p = 1, 2, ..., Mamba\ layers$ **do**
  $\mathbf{A}$: (V, N) $\leftarrow$ Parameter
  $\mathbf{B}$: (B, L, N) $\leftarrow s_B(\mathbf{X}')$
  $\mathbf{C}$: (B, L, N) $\leftarrow s_C(\mathbf{X}')$
  Δ: (B, L, N) $\leftarrow \tau_\Delta$(Parameter + $s_\Delta(\mathbf{X}')$)
  $\bar{\mathbf{A}}, \bar{\mathbf{B}}$: (B, L, V, N) $\leftarrow discretize(\Delta, \mathbf{A}, \mathbf{B})$
  $U'^{(1)} \leftarrow \text{SSM}\ (\bar{\mathbf{A}}, \bar{\mathbf{B}}, \mathbf{C})(\mathbf{X}')$
  $U'^{(2)} \leftarrow U'^{(1)} \otimes \text{SiLU}(Linear(\mathbf{X}'))$
  $U' \leftarrow Linear(U'^{(2)})$
end
return U';

6.2 PRELIMINARY

Mamba. Mamba is proposed in (Gu & Dao, 2023). With four parameters $\mathbf{A}, \mathbf{B}, \mathbf{C}, \Delta$, Mamba is defined based on a sequence-to-sequence transformation via the following equations:

$$\begin{aligned} h'(t) &= \mathbf{A}h(t) + \mathbf{B}x(t); \\ y(t) &= \mathbf{C}h(t); \\ h_t &= \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t \end{aligned} \tag{9}$$

where $h(t)$ denotes the hidden state, $x(t)$ is the input sequence, $y(t)$ is the output sequence, and $\mathbf{A} \in \mathbb{R}^{N\times N}, \mathbf{B} \in \mathbb{R}^{L\times N}, \mathbf{C} \in \mathbb{R}^{N\times L}$. In addition, N and L are the dimension factor and the sequence length, respectively. The discretization process of parameters (A, B) is shown as follows:

$$\bar{\mathbf{A}} = \exp(\Delta\mathbf{A}); \quad \bar{\mathbf{B}} = \Delta\mathbf{A}^{-1}\exp(\Delta\mathbf{A}) \cdot \Delta\mathbf{B} \tag{10}$$

Here the discretization is closely related to continuous-time systems, providing them with additional properties such as resolution invariance (Nguyen et al., 2022) and automatic normalization, ensuring the model's proper calibration. Mamba achieves input-dependent selection by making $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{\Delta}$ functions of the input x. In this way, Mamba is able to dynamically adjust its operations, computations, and information flow based on the specific characteristics of the input data. This input-dependent selection allows Mamba to effectively adapt its behavior and capture the relevant patterns and dynamics present in the input, resulting in enhanced modeling capabilities and improved performance for various tasks. Then a state-space model (SSM) utilize $\bar{\mathbf{A}}, \bar{\mathbf{B}}$, and $\mathbf{C}$ to process the input x:

$$\bar{\mathbf{K}} = (\mathbf{C}\bar{\mathbf{B}}, \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}}, ...\mathbf{C}\bar{\mathbf{A}}^k\bar{\mathbf{B}}, ...)^T, \;\; y = \bar{\mathbf{K}}^T x \tag{11}$$

Finally, the output y of the SSM is multiplied with a non-linear activation-transformed input. This result is then passed through a final linear layer to produce Mamba's output. For a complete overview of Mamba's architecture, refer to Algorithm 3.

Fourier Transform. Given the input function $\mathbf{f}(x)$, we can obtain the frequency domain conversion function $\mathbf{F}(k)$ via the Discrete Fourier Transform (DFT), where $\mathbf{F}$ denotes the Fourier transform of the function $\mathbf{f}(x)$. The process is shown as follows:

$$\begin{aligned} \mathbf{F}(k) &= \int_d \mathbf{f}(x) e^{-j2\pi kx} dx \\ &= \int_d \mathbf{f}(x)\cos(2\pi kx)dx + j\int_d \mathbf{f}(x)\sin(2\pi kx)dx \end{aligned} \tag{12}$$

In this context, we have the frequency variable denoted as k, the spatial variable as x, and the imaginary unit as j. The real part of $\mathbf{F}$ is represented as $\text{Re}(\mathbf{F})$, while the imaginary part is denoted as $\text{Im}(\mathbf{F})$. The complete conversion is expressed as $\mathbf{F} = \text{Re}(\mathbf{F}) + j\text{Im}(\mathbf{F})$. The Fourier transform is employed to decompose the input signal into its constituent frequencies. This process facilitates the identification and detection of periodic or aperiodic patterns, which are crucial for tasks such as image recognition.

Laplace Transform. The Laplace transform is a powerful mathematical tool used in various fields, particularly in engineering, physics, and applied mathematics. It allows us to convert functions of time into functions of complex variables, providing a useful way to analyze and solve differential equations. The Laplace transform of a function, denoted as $F(s)$, is defined as follows:

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t)\, dt \tag{13}$$

In this equation, $f(t)$ is the original function in the time domain, s is a complex variable, and $F(s)$ is the transformed function in the complex frequency domain. The Laplace transform has several important properties that make it a versatile tool for analysis. For example, it enables us to simplify differential equations into algebraic equations, making it easier to solve for unknown functions. Additionally, the Laplace transform allows us to study system behavior, stability, and response to different inputs. By applying the inverse Laplace transform, we can obtain the original function back from its transformed representation. This transformation provides a valuable method

for understanding and manipulating functions in the frequency domain, facilitating analysis and design in various scientific and engineering disciplines.

The inverse Laplace transform is defined as follows:

$$f(t) = \mathcal{L}^{-1}\{F(s)\} = \lim_{T\to\infty} \int_{\gamma-iT}^{\gamma+iT} e^{st} F(s)\, ds \tag{14}$$

Here $Re(s) = \gamma$ and γ is greater than the real part of all singularities of $F(s)$. For general functions, the inverse Laplace transform may not have analytical solution.

To allow analytical solution for inverse Laplace transform, we follow (Cao et al., 2023) and consider a neural operator

$$u(t) = \sigma((\kappa(\mathbf{f};\phi) * v)(t) + \mathbf{W}v(t)) \tag{15}$$

where σ is a nonlinear activation function, $\mathbf{W}$ is a linear transformation, and κ is a kernel integral transformation. Imposing $\kappa_\phi(t,\tau) = \kappa_\phi(t-\tau)$, in the Laplace space we have

$$U(s) = K_\phi(s)V(s) \tag{16}$$

where $K_\phi(s) = \mathcal{L}\{\kappa_\phi(t)\}$ and $V(s) = \mathcal{L}\{v(t)\}, U(s) = \mathcal{L}\{u(t)\}$.

Here we assume that the kernel integral operator has the form of $K_\phi(s) = \sum_{n=1}^{N} \frac{\beta_n}{s-\mu_n}$ in the Laplace space, where $\beta_n \in \mathbb{R}$ and $\mu_n \in \mathbb{C}$ are learnable parameters. We make the assumption so that the singularities are first-order, and the inverse Laplace transform has analytical solution. After some derivation, we have that the resulting form for $u(t)$ in the original space is

$$u(t) = \sum_{n=1}^{N} \gamma_n \exp(\mu_n t) + \sum_{l=-\infty}^{\infty} \lambda_l \exp(i\omega_l t) \tag{17}$$

Here ω_l are frequencies by decomposing $v(t)$ via Fourier series, and γ_n, λ_l are derived parameters from β_n, ω_l and μ_n. For detailed derivation, see (Cao et al., 2023).

Here we see that by assuming the first-order singularities for the kernel where the singularities positions μ_n are learnable, we can parameterize both the decay and periodic behavior in the original space $\sum_{n=1}^{N} \gamma_n \exp(\mu_n t)$, via μ_n's real and imagery part, respectively. We further condition the real and imagery part, as well as its amplitude γ_n on the output of the previous Mamba modules in our FLDmamba, allowing the model to decide appropriate decay and periodic dynamics depending on the input.

6.3 Model Complexity

This section presents a complexity analysis of our proposed model, FLDmamba. The computational complexity of the base Mamba model is $\mathcal{O}(BLVN)$, where B represents the batch size, L denotes the sequence length, V signifies the number of variables, and N indicates the state expansion factor. The Fast Fourier Transform (FFT) in FLDmamba has time complexity of $\mathcal{O}(BLN \log L)$, and the inverse Laplace transform has time complexity of $\mathcal{O}(BLN)$, both significantly smaller than $\mathcal{O}(BLVN)$. Therefore, the total time complexity is still $\mathcal{O}(BLVN)$. In other words, FLDmamba maintains a comparable computational time complexity to the base Mamba model, making it a promising framework for large-scale real-world applications in time series prediction. This computational efficiency allows FLDmamba to handle extensive datasets and complex time series scenarios without significant performance degradation.

6.4 EXPERIMENTS

6.4.1 EXPERIMENT SETTINGS

To ensure a fair comparison, we modify the hidden dimensionality of all compared algorithms within the range of $[128, 256, 512, 1024, 2048]$ to achieve their reported best performance, which is consistently observed at 1024. The learning rate (η) is initialized to 5×10^{-6}, and we set the number of FLDmamba layers to 2. Consistent with the existing settings of time series datasets, we utilize historical data with 96, 192, 336, or 720 time steps. The time steps are defined as 5 minutes, 1 hour, 10 minutes, or 1 day intervals to predict the corresponding future 96, 192, 336, or 720 time steps in these time series datasets. All baseline methods are evaluated using their predefined settings as described in their respective publications. We conduct testing for all tasks on a single NVIDIA L40 GPU equipped with 128 CPUs.

Table 2: The statistics of 9 public datasets.

Datasets	Variates	Timesteps	Granularity
ETTh1&ETTh2	7	69,680	1 hour
PEMS04	307	16,992	5 minutes
PEMS08	170	17,856	5 minutes
Exchange	8	7,588	1 day
Electricity	321	26,304	1 hour
Solar-Energy	137	52,560	10 minutes
ETTm1&ETTm2	7	17,420	15min

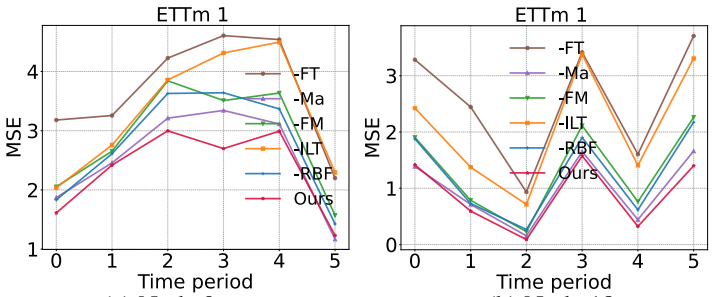
6.4.2 BASELINE DESCRIPTIONS

Transformer-based methods:

- Autoformer (Wu et al., 2021) employs a series decomposition technique along with an Auto-Correlation mechanism to effectively capture cross-time dependencies.
- FEDformer (Zhou et al., 2022) introduces an enhanced Transformer operating in the frequency domain, aiming to improve both efficiency and effectiveness.
- Crossformer (Zhang & Yan, 2022) incorporates a patching operation like other models but distinguishes itself by employing Cross-Dimension attention to capture dependencies between different series. While patching reduces the elements to process and extracts semantic information comprehensively, these models encounter performance limitations when handling longer.
- DLinear (Zeng et al., 2023) introduced DLinear, a method that decomposes time series into two distinct components and generates a single Linear layer for each component. This straightforward design has outperformed all previously proposed complex transformer models.
- PatchTST (Huang et al., 2024) leverages patching and channel-independent techniques to facilitate the extraction of semantic information from single time steps to multiple time steps within time series data.
- iTransFormer (Liu et al., 2023) employs inverted attention layers to effectively capture inter-series dependencies. However, its tokenization approach, which involves passing the entire sequence through a Multilayer Perceptron (MLP) layer, falls short in capturing the complex evolutionary patterns inherent in time series data.

MLP-based methods:

- TimesNet (Wu et al., 2022) expands the examination of temporal fluctuations by extending the 1-D time series into a collection of 2-D tensors across multiple periods.
- RLinear (Li et al., 2023), the state-of-the-art linear model, incorporates reversible normalization and channel independence into a purely linear structure.



(a) Node 9 (b) Node 18

Figure 8: Ablation study of FLDmamba on prediction performance on Node 9 and Node 18 instances of ETTm 1 dataset.

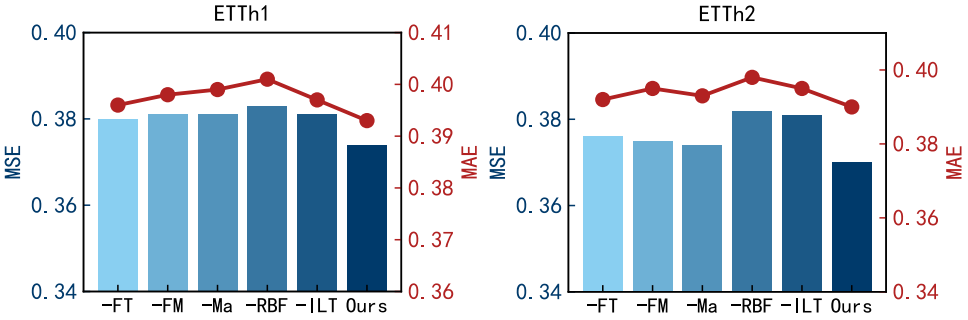


Figure 9: Ablation study of FLDmamba on four datasets with $L = 96$.

- TiDE (Das et al., 2023) is an encoder-decoder model that employs a Multi-layer Perceptron (MLP) architecture.

SSM-based methods:

- S-Mamba (Wang et al., 2024) independently tokenizes the time points for each variate using a linear layer. This allows for the extraction of correlations between variates using a bidirectional Mamba layer, while a Feed-Forward Network is employed to learn temporal dependencies.

6.5 Efficiency (Q6)

This section evaluates the computational efficiency of our proposed framework, FLDmamba, in comparison to several state-of-the-art baselines, including AutoFormer, RLinear, iTransformer, and S-Mamba. We assess efficiency on the ETTh1 and ETTh2 datasets, considering both training time per epoch and GPU memory consumption. The results, presented in Figure 12, demonstrate the following: **Comparative Efficiency of FLDmamba:** Our method, FLDmamba, exhibits a favorable balance between performance and computational efficiency, achieving comparable training times and GPU memory costs to baselines. **Efficiency of Mamba-Based Methods:** Mamba-based methods, including FLDmamba and S-Mamba, demonstrate a compelling advantage in terms of training

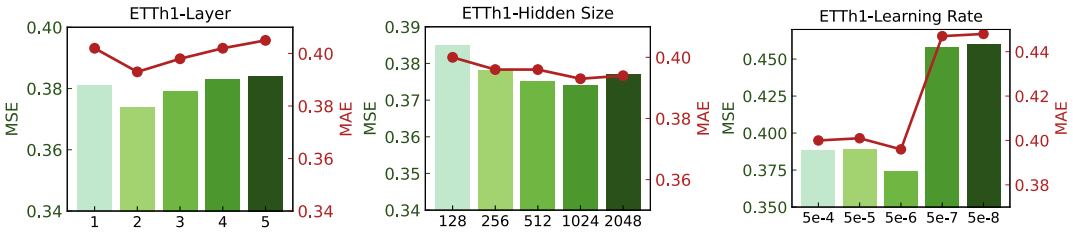


Figure 10: Hyperparameter study of FLDmamba.

time and GPU memory consumption compared to Transformer-based baselines such as AutoFormer. This suggests that Mamba-based architectures offer a more efficient approach for handling time series data. These findings highlight the computational efficiency of our proposed framework, FLDmamba, while also emphasizing the potential benefits of Mamba-based architectures for addressing computational resource constraints in time series modeling.

6.5.1 HYPERPARAMETER STUDY (Q7)

In this section, we aim to conduct a parameter study to evaluate the impact of important parameters on the performance of our model, FLDmamba. The results are presented in Figure 10. Specifically, we vary the number of FLDmamba layers within the range of $\{1, 2, 3, 4, 5\}$, the hidden size from $\{128, 256, 512, 1024, 2048\}$, and the learning rate from $\{5 \times 10^{-4}, 5 \times 10^{-5}, 5 \times 10^{-6}, 5 \times 10^{-7}, 5 \times 10^{-8}\}$. Based on the results, we provide a summary of observations regarding these three parameters and their effects on performance, measured by MSE and MAE metrics, as follows: **(1)** We examine the impact of FLDmamba layers on the performance of FLDmamba. We observe that FLDmamba achieves the best performance when the number of layers is set to 2. However, as we increase the number of FLDmamba layers, the performance starts to diminish. This suggests that additional layers may introduce an over-smoothing effect, which negatively affects the performance of FLDmamba. **(2)** We also conducted experiments to investigate the effect of hidden sizes on FLDmamba performance. We find that our model FLDmamba achieves the highest performance when the hidden size is set to 1024. This indicates that smaller hidden sizes may not provide sufficient information, while larger hidden sizes may introduce redundant information that hampers the performance of FLDmamba. **(3)** Furthermore, we examine the impact of the learning rate on performance and observe that our method FLDmamba achieves the best performance when the learning rate is set to 5×10^{-6}. Smaller or larger learning rates may result in insufficient convergence or overfitting, which adversely affects the performance.

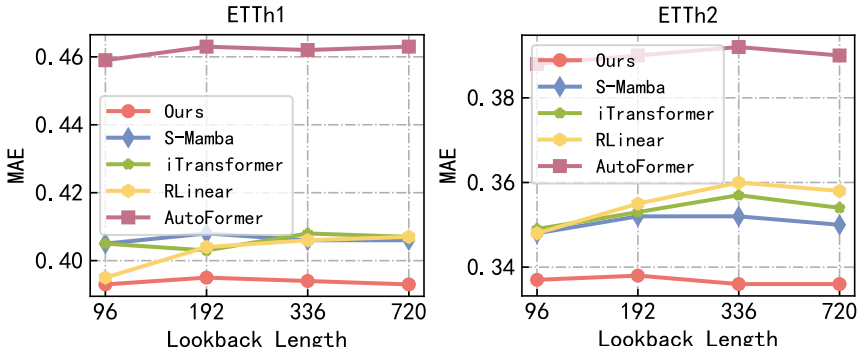


Figure 11: Long-term prediction with the lookback length from the range [96, 192, 336, 720].

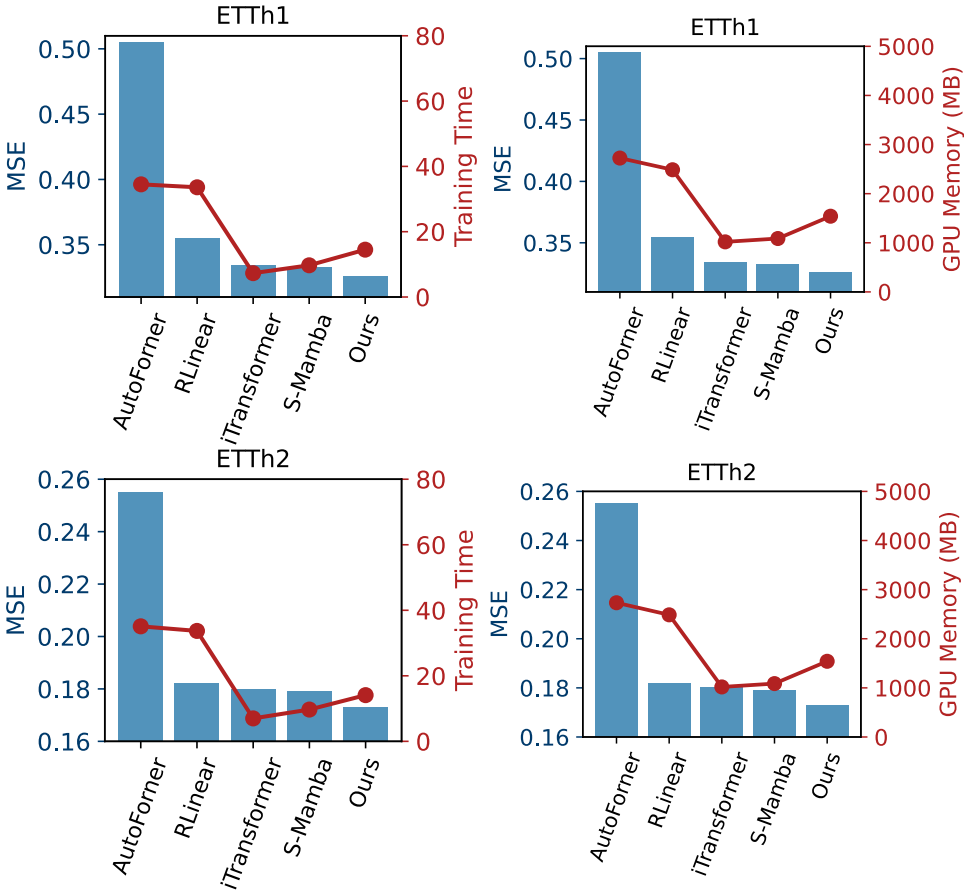


Figure 12: Model efficiency comparison on ETTh1 and ETTh2. The batch size is 32.

6.6 Limitations and Future Work

The limitation of our work involves potential challenges in scaling the proposed model to extremely large datasets. Future efforts will focus on improving the model's adaptability to dynamic data environments and assessing its performance across diverse time series datasets. Furthermore, the exploration of alternative kernel functions beyond the RBF and a thorough scalability analysis will be pursued. Lastly, extending the model to accommodate missing data and integrating uncertainty quantification in predictions will bolster its practical utility.

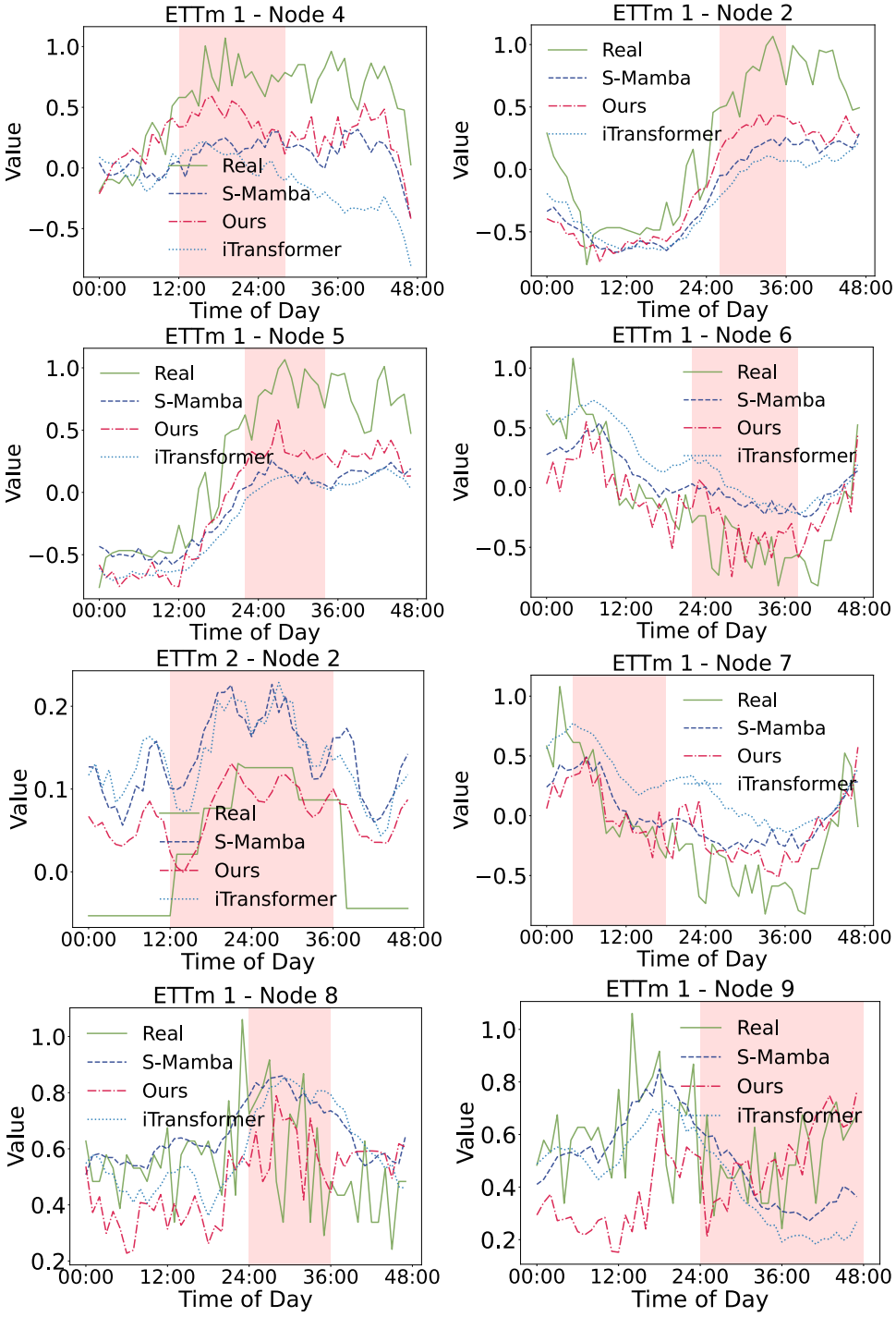


Figure 13: Case study of FLDmamba in terms of multi-scale periodicity.